論文94-31B-11-6

파이프라인 방식의 버스를 위한 비 동기식 주 기억장치의 설계 및 구현

(Design and Implementation of Asynchronous Memory for Pipelined Bus)

韓 宇 宗*, 金 壽 遠**

(Woo-Jong Hahn and Soo-Won Kim)

要 約

최근 고성능 마이크로 프로세서들의 가격 경쟁력에 힘입어 공유 버스 방식의 다중 처리기 시스템이 많이 등장하고 있다. 이들 다중 처리기 시스템들은 주기억장치의 구조에 따라 성능이 크게 달라질 수 있다. 주기억장치의 중요성은 마이크로 프로세서들이 고속화 되어감에 따라 더욱 커지고 있다. 개개의 마이크로 프로세서들을 위한 캐시 메모리가 대부분의 시스템에서 채용되고 있으나 여전히 공유되는 주기억장치의 접근 특성은 다중 처리기 시스템의 성능과 확장성을 제약하는 요소가 된다.

본 논문에서는 파이프라인 방식의 시스템 버스의 효율성을 최대한 유지하면서 주기억장치 구현의 유연성을 제공하는 비동기적 주기억장치의 구조를 제안하며 그 효과를 시뮬레이션을 통하여 보이고 있다. 시스템 버스로는 고속 중형 컴퓨터를 위하여 설계된 HiPi+Bus를 모델로 하고 있으며 Verilog를 이용하여 시뮬레이션 하였다. 이 시뮬레이션을 통하여 제안된 비동기적 주기억장치 구조가 시스템 버스의 사용률을 낮추어 줌으로써 시스템의 성능과 확장성을 향상시킴을 알 수 있었다. 또한 제안된 구조를 구현하기 위한 구현 방법상의 변수들을 평가하였으며 구현된 주기억장치를 시험 프로그램을 이용한 시험 환경에서 시험하여 그 동작과 유용성을 확인하였다.

Abstract

In recent days low cost, high performance microprocessors have led to construction of medium scale shared memory multiprocessor systems with shared bus. Such multiprocessor systems are heavily influenced by the structures of memory systems and memory systems become more important factor in design space as microprocessors are getting faster. Even though local cache memories are very common for such systems, the latency on access to the shared memory limits throughput and scalability. There have been many researches on the memory structure for multiprocessor systems.

In this paper, an asynchronous memory architecture is proposed to utilize the bandwith of system bus effectively as well as to provide flexibility of implementation. The effect of the proposed architecture is shown by simulation. We choose, as our model of the shared bus is HiPi+Bus which is designed by ETRI to meet the requirement of the High-Speed Midrange Computer System. The simulation is done by using Verilog hardware decription language. With this simulation, it is explored that the proposed asynchronous memory architecture keeps the utilization of system bus low enough to provide better throughput and scalibility. The implementation trade-offs are also described in this paper. The asynchronous memory is implemented and tested under the prototype testing environment by using test program. This intensive test has validated the operation of the proposed architecture.

* 正會員, 韓國電子通信研究所
(Elec. and Telecommunications Research Institute)
** 正會員, 高麗大學校 電子工學科
(Dept. of Elec. Eng., Korea Univ.)

接受日字 : 1994年 7月 26日
※ 본 연구는 고속중형 컴퓨터 개발 사업의 일부로 이루어졌음

Ⅰ. 서 론

최근의 마이크로 프로세서의 기술 발전에 힘입어 상용의 고성능 마이크로 프로세서들을 수 개에서 수십개까지 공유 버스에 연결한 대칭형 다중 처리기 시스템들이 많이 등장하고 있다.[1][2][3] 이들 다중 처리기 시스템들은 가격대 성능비의 우월성을 바탕으로 중형 컴퓨터 분야를 주도하고 있고, 최근에는 SUN사와 같은 워크스테이션 업체나 CRAY Research사등도 자회사를 통하여 공유 버스 방식의 다중 처리기를 내놓고 있다.

그림 1에 일반적인 다중처리기 시스템을 나타내었다. 그림 1에서 나타나듯이 다중 처리기 시스템들은 공유 버스에 연결된 하나 이상의 블록으로 구성되는 공유 주기억장치를 제공하고 있다.

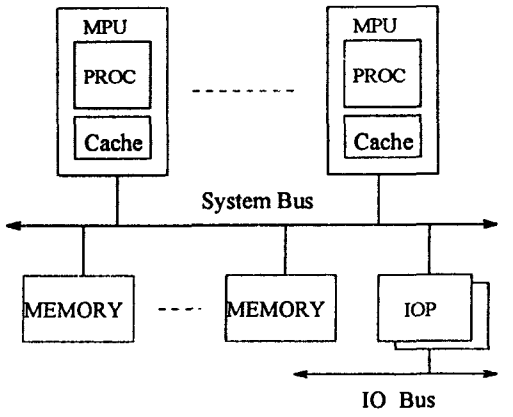

그림 1. 다중 처리기의 구조
Fig. 1. Architecture of Multiprocessor.

공유 주기억장치는 위와 같은 다중처리기 시스템에서 주 처리장치나 입출력 장치등의 여러 요구기에 대한 응답을 해야 하는 유일한 응답기로서 그 구조 및 설계가 시스템의 성능과 확장성에 큰 영향을 미친다. 단일 처리기 시스템과는 달리 다중 처리기 시스템에서의 주기억장치 설계시 시스템 버스의 사용률이 매우 중요한 변수이다. 이것은 사용률이 증가하면 시스템 버스의 사용 요구간 충돌이 증가하고 결국 접근시간의 심각한 증가를 가져온다. 이때 시스템 버스상의 충돌로 인한 지연 시간은 주기억장치의 접근시간 자체보다도 클 수 있다. 기억장치의 계층화에도 불구하고 프로세서의 속도가 매우 빠른 속도로 높아지고 있으므로 위와같은 주기억장치 구조 설계상의 변수는 더욱 중요해지고 있다.

기본적으로 프로세서 속도 증가에 따른 시스템 버스 사용률의 증가 문제를 해결하기 위한 방법을 세가지로 분류할 수 있다. 첫째, 시스템 버스의 데이터 폭을 늘리고 동작 주기는 줄이는 방법을 생각할 수 있다. 이 방법은 근본적인 대책은 될 수 있으나 백플레인 구성시 길이나 전도체의 재질등에 따른 물리적 한계가 있으므로 이 방법만으로는 효과가 매우 제한적이다. 두번째 방법으로 시스템 버스로의 요구 자체를 줄이는 것을 생각할 수 있다. 각 프로세서를 위한 지역 캐시 메모리가 이 경우 매우 효과적이고 대부분의 다중 처리기 시스템에 사용되고 있다. 프로세서의 속도가 매우 빠른 속도로 발전하고 있으므로 캐시의 적중률이 100 %가 아닌 한 마지막으로, 주기억장치를 효과적으로 구성하여 재시도를 줄임으로서 시스템 버스의 사용률을 줄이는 방법이 있다. 이들은 어느 한가지 방법만으로는 시스템 버스상의 충돌을 효과적으로 제한할 수 없으므로 복합되어 사용되어야 한다.

기억장치의 구조에 관한 많은 연구가 있었으나 상당수가 기억장치의 계층구조에 관한 것으로 지역 캐시 메모리를 대상으로 한 것들이다.[4][5][6] 해석적 방법에 의한 전체적 성능 분석을 한 연구도 있으나[7] 공유 기억장치 구조의 설계 변수에 중점을 두지 않고 있다. 계층적 기억장치 구조에 관한 시뮬레이션을 통하여 부분적으로 공유 기억장치의 입력 버퍼의 유용성을 보인 연구 결과도 있었다.[8] 본 논문은 시스템 버스를 통하여 공유되는 주기억장치의 구성에 있어서 입출력 큐를 두어 동기식인 파이프라인 버스와 비 동기적으로 동작하도록 하여 주기억장치가 서비스할 수 없는 상태인 경우에도 시스템 버스상의 재시도를 방지하여 사용률을 제한하는 방법에 관한 것이다. 즉, 주기억장치가 다른 처리기에 의하여 사용되고 있을 때 또 다른 처리기로 부터의 요구가 있는 경우 이를 재시도하게 하지 않고 입력 큐를 이용하여 받아들임으로써 시스템 버스의 대역폭을 유용하게 쓸 수 있도록 한 것이며 주기억장치가 응답을 하는 경우에도 출력 큐를 이용하여 같은 효과를 내도록 하는 것이다.

본 논문은 5개의 장으로 되어 있으며 서론에 이어 모델이 되는 시스템 버스의 동작이 2장에서 기술되었고 3장에서는 비 동기식 주기억장치의 구조 및 평가가 기술되었다. 구현에 관련된 변수들과 구현된 모습은 4장에서 설명되며 마지막으로 결론이 5장에 나타나 있다.

Ⅱ. Hipi+Bus

그림 1과 같은 공유 버스를 갖는 다중 처리기 시스템에서는 공유 버스가 병목점이 될 수 있다. 수개이

상의 프로세서를 연결하기 위해서는 공유 버스가 높은 대역폭과 아울러 매우 효율적인 프로토콜을 가져야 한다.

본 논문에서 모델로하고 있는 HiPi+Bus는 TICOM에서 사용되는 HiPi-Bus를 개량한 것으로 BTL(Bus Transceiver Logic)을 채용하여 사이클 주기를 25 % 줄인 60 ns의 주기를 갖는 동기형 버스로 최대 264 MByte/sec의 전송 대역폭을 지원한다. 기본 프로토콜은 HiPi-Bus와 유사하며 높은 확장성을 위하여 이 버스가 갖는 특징은 다음과 같다.[9]

1. 중재 기능

시스템 구성의 유연성을 가지면서 신뢰성을 높이기 위하여 분산 중재 방식을 제공하고 있으며 각 블록에 대하여 하나의 비트를 할당하는 선형 자가 중재 방식을 쓰고 있다. 또한 기본적으로는 우선순위에 의한 중재를 하나 우선순위가 낮은 블록의 굶주림 현상을 방지하기 위하여 공정성 있는 중재 방식을 제공하고 있다.e 이 기능은 각 블록에서 자발적으로 구현하는 것이므로 특별히 계속 버스의 사용권을 획득하는 것이 시스템 전체적인 측면에서 유리하다고 판단되는 경우 해당 블록은 이 기능을 사용하지 않을 수도 있다.

HiPI+Bus는 HiPI-Bus와 같이 데이터 요구와 그 응답이 서로 독립적으로 버스상에 나타날 수 있는 분리된 동작(split transaction)을 지원하므로 주기억장치가 읽기 요구에 대한 응답을 할 때도 버스의 중재가 필요하며 그 기능은 위에서 설명한 기본 중재와 거의 동일하다. 이와같이 분리된 동작을 지원함으로써 한 블록이 불필요하게 버스를 홀드하는 시간을 없애고 보다 많은 블록의 연결이 가능하다.

2. 데이터 전송

데이터 전송 기능은 주소 정보와 데이터 정보를 버스상에서 블록간에 전달하는 것을 기본 기능으로 하며 주소 정보로는 시스템 주소표상의 주소 지정 정보 외에도 주소 공간 지정, 전송 형태 및 태그 정보를 포함한다. 앞에서 기술한 바와 같이 데이터 요구시 분리된 동작을 지원하므로 주기억장치가 데이터 응답시 요구 블록을 알 수 있도록 요구 블록의 태그 정보를 주소 정보 전송시 같이 전송하도록 한다. 전송 형태로는 모두 13가지가 정의되어 있으며 그 내용은 표 1과 같다.

주기억장치의 입장에서는 표 1의 전송 형태들은 크게 4가지로 분류할 수 있다. 즉, 단일 전송 형태의 읽기와 쓰기, 블록 전송형태의 읽기와 쓰기로서 NRD, NCR, ILR, NCW, NWR, ILW등이 전자에 속하고 BRD, CRD, EXR, BWR, WRB, ITV 등이 후자로 분류된다. 단일 전송 형태와 블록 전송 형태의 동작을 그림 2와 그림 3에 나타내었다.

표 1. Hipi+Bus의 전송형태
Table 1. Transfer Types of HiPi+Bus.

Mnemonics	name	category	cache-related
NRD	normal read	single read	yes
NWR	normal write	single read	yes
NCR	non-coherent read	single read	no
NCW	non-coherent write	single read	no
IVD	invalidation		yes
ILR	interlock read	single read	yes
ILW	interlock write	single write	yes
BRD	block read	quadruple read	yes
BWR	block write	quadruple write	yes
CRD	coherent read	quadruple read	yes
EXR	exclusive read	quadruple read	yes
WRB	write-back	quadruple write	yes
ITV	intervention write	quadruple write	yes

그림 2는 HiPi-Bus의 전송 형태와 같은 것으로 버스를 사용하고자 하는 블록은 중재 사이클동안 중재에 참가하여 사용권을 획득하고 주소 정보와 쓰기시의 데이터 정보를 전송한다. 이때 주기억장치는 주소 정보를 번역하여 자신이 응답해야 하는지, 또 읽기인지 쓰기인지도 판단하여 자신에 대한 쓰기인 경우 데이터를 버스로 부터 받아 저장한다. 이때 주소 정보를 받고 자신에 대한 요구인 경우 반드시 한 사이클 후에 응답 신호를 보내야 한다. 이것은 요구한 블록이 버스 사용을 재시도할 것인지 판단하기 위하여 필요하며 이러한 동작들은 모두 클럭에 동기되어 일어난다. 읽기 요구인 경우 요구 블록은 주소 정보 전송 후에 버스의 사용권을 내놓으며 주기억장치는 내부에서 읽기를 마친 후 다시 버스의 사용권을 획득하여 데이터를 전송한다.

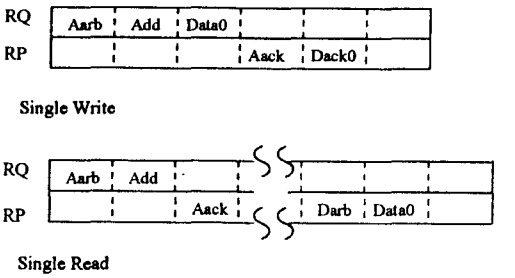

그림 2. 단일 전송 형태의 읽기와 쓰기
Fig. 2. Single Read and Write.

HiPi+Bus는 데이터 전송 측면에서 크게 두가지가 개선되어 16 byte의 데이터 폭을 지원하고 64 byte

의 블록 전송을 지원한다. 이와 같은 블록 전송은 각 지역 캐시 메모리가 커짐에 따라 보다 큰 캐시 라인을 지원할 필요가 있기 때문이다.

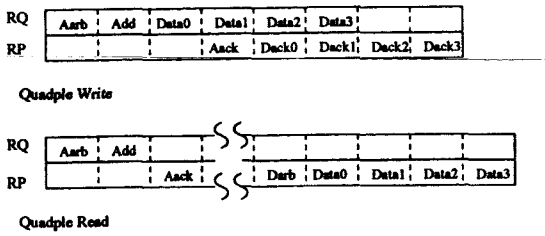

그림 3. 블록 전송 형태의 읽기와 쓰기
Fig. 3. Quadruple Read and Write.

그림 3에 나타난 바와 같이 블록 전송 형태는 기본적으로 단일 전송 형태와 같은 방식이며 단지 데이터 사이클이 확장되었다. 그러나 HiPi+Bus와 같은 동기식 버스에서 확장된 데이터 사이클은 데이터 버스의 충돌을 막기 위한 파이프라인상의 제어가 필요하며 이를 위하여 HiPi-Bus보다 몇가지 신호가 추가되었으며 주기억장치의 상태 제어기도 기능이 추가되었다. 그러나 보다 심각한 잠재적 문제는 데이터 사이클이 주소 정보 사이클보다 확장됨으로서 파이프라인의 균형이 깨지고 있다는 것으로 이로 인하여 블록들 간의 버스 사용상의 불균형이 생길 수 있다.[10] 본 논문에서 제안하는 비 동기식 주기억장치의 구조는 뒤에서 설명하는 바와 같이 이러한 불균형을 시정할 수 있다.

Ⅲ. 비동기식 주기억장치

많은 처리기를 효과적으로 지원하기 위한 팬디드, 파이프라인 방식의 시스템 버스인 HiPi+Bus와 인터페이스되면서 주기억장치 내부 구성에 유연성을 가질 수 있는 비동기식 주기억장치의 세부 구성은 그림 4와 같다.

비동기식 주기억장치 블록은 10개의 내부 기능 모듈로 구성되며 이들 각각의 동작은 다음장에서 기술할 것이다. 제안된 그림 4와 같은 구조의 비동기식 주기억장치는 다음과 같은 장점을 가진다.

첫째, 시스템 버스가 보다 많은 처리기와 IO 요구 등을 효과적으로 처리할 수 있도록 개선되어 갈때 상대적으로 느린 소자를 사용하는 주기억장치의 동작에 크게 영향받지 않을 수 있다. 주기억장치는 DRAM의 사용이 필수적이며 DRAM의 속도는 프로세서는 물론 BTL 또는 FAST TTL 기술로 구현 되는 시스템 버스의 속도보다도 느리므로 이 점은 성능 향상에 있어서 중요한 점이라고 할 수 있다. 이점은 대용량 주기억장치가 EDC(Error Detection & Correction) 기능을 갖는다고 볼때 더욱 중요하다.

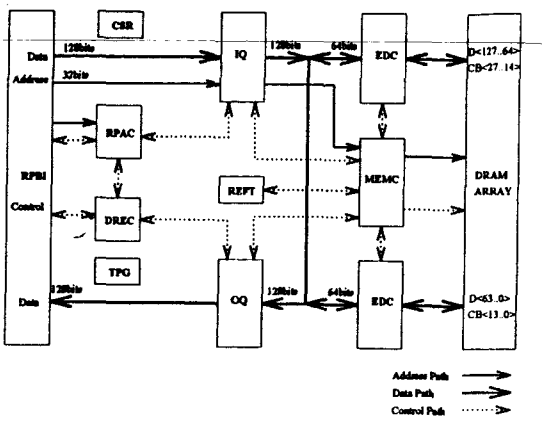

그림 4. 비동기식 주기억장치의 구성
Fig. 4. Block Diagram of Asynchronous Memory.

둘째, 파이프라인 방식의 시스템 버스를 상대적으로 느린 주기억장치가 따라갈 수 있는 버퍼를 제공하므로 시스템 버스에서 재시도로 인한 사용률의 증가를 막을 수 있다. 시스템 버스가 처리기들의 요구를 파이프라인 방식으로 효과적으로 전달하더라도 처리 속도가 느린 주기억장치가 그 요구들을 바로 처리할 수 없으므로 한 요구에 대한 처리가 진행 중일때 다른 요구들을 수신하지 않게 되고 이들은 처리기에서 재시도되므로 결국 처리기의 수와 속도가 증가하면서 시스템 버스상의 재시도도 증가하게된다. 제안된 비동기식 주기억장치는 이러한 불필요한 재시도를 제거할 수 있고 결과적으로 시스템 버스의 대역 폭을 늘리는 효과를 가져와 확장성을 향상시킨다.

세째, 주기억장치에 대한 요구가 재시도되지 않고 주기억장치에서 대기하게 되므로 결과적으로 접근시간을 줄일 수 있다. 즉, 재시도를 하게되면 재시도 주기의 배수만큼 대기하게 되나 본 방식에서는 꼭 필요한 최소 시간만 대기한다.

네째, 주기억장치의 입력과 출력 부분이 분리되므로 주기억장치가 데이터 버스의 사용권을 확보하지 못하여 대기상태에 들어가더라도 처리기의 요구를 받아 처리해 줄 수 있다. 이 점도 데이터 버스의 사용량이 많거나 주기억장치의 우선순위가 낮거나 또는 블록 데이터 전송등으로 인하여 데이터 버스의 사용이 지연될 때 그 영향으로 주기억장치가 처리기의 요

구를 재시도시키는 것을 방지해 준다.

다섯째, 블록 데이터 전송으로 인한 특정 처리기의 굶주림을 방지할 수 있다. 이 점은 뒤의 시뮬레이션 결과에서 다시 설명될 것이다.

마지막으로, DRAM 소자의 발전이나 가격의 변화 등에 따른 주 기억장치의 내부 구성의 변경이 용이하다는 점이다. 즉, 접근 시간이 다른 DRAM을 사용한다거나, 제어회로의 구현상의 변경을 하고자 할 때 시스템 버스 동작의 복잡한 경우를 고려할 필요가 없다.

그림 4의 주기억장치가 처리해야할 동작들은 다음과 같다.

단일 데이터 읽기
블록 데이터 읽기
단일 데이터 쓰기
블록 데이터 쓰기
부분 데이터 쓰기
상태 및 제어 레지스터 읽기
상태 및 제어 레지스터 쓰기
리프레시
ECC(Error Checking Code) 초기화

위에 열거된 동작들 중 ECC 초기화 동작은 주기억장치의 초기화시에 자동으로 수행되는 것으로 ECC를 생성하고 ECC용 DRAM에 기록하는 일이다. 이 과정이 없을 경우 처리기들로 부터 주기억장치에 대한 읽기 요구가 있을 때 ECC 오류로 응답하게 될것이다.

비동기식 주기억장치의 효과를 알아보기 위하여 Verilog를 이용하여 시뮬레이션 모델을 작성하였다. Verilog는 실제 블록의 설계 및 논리 시뮬레이션에도 사용된 것으로 이 언어를 같이 사용함으로써 대상을 실제 구성시와 근접하게 모델링할 수 있었다. 일반적으로 Verilog는 실제 구현을 위한 회로 설계시에 많이 사용되는 것으로 구조의 모델링이 가능하다고 하나 작성에 많은 시간이 소요되고 비교적 고가의 CAD환경이 수반되어야 한다는 단점이 있어 모델링에는 많이 사용되지 않고 있다. 그러나 본 연구에서는 실제 구현을 염두에 두고 시작하였으므로 모델링의 결과가 최소한의 변경으로 실제 구현 회로의 검증용으로 활용될 수 있고 CAD 환경도 이미 갖추어진 환경을 이용하는 것이므로 최대한 실제 구현 내용을 반영할 수 있도록 Verilog를 사용하였다. 처리기와 주기억장치를 모델링하였으며 이때 처리기는 분석의 대상이 아니므로 시스템 버스 인터페이스와 주소발생기만을 갖는 단순화된 것이다. 16개의 처리기와 4개의 주기억장치가 연결되어 있을때 throughput에 미치는 영향을 그림 5에 나타내었다. 단, 이때 주소 패턴은 난수발생을 이용한 것이며 throughput이란 시스템 버스상의 throughput을 의미하는 것으로 처리기의 성능등은 제외된 것이다.

그림 5에서 최소 throughput을 보인 처리기와 최고 throughput을 보인 처리기간의 차이를 분석함으로써 처리기간의 시스템 버스 획득의 불균형을 비동기식 주기억장치가 해결할 수 있음을 보이고 있다. 즉, 입출력에 충분한 큐를 제공함으로서 블록 전송으로 인해 파이프라인의 단계간 균형이 깨지고 이로 인한여 시스템 버스의 유효 대역폭이 줄어드는 부작용이 생기는 문제를 흡수하여 특정 처리기의 굶주림 현상을 방지할 수 있다. 이와 같은 불균형은 주기억장치의 수가 적을수록, 처리기의 수가 많을수록, 프로세서의 속도가 빠를수록 심해진다.

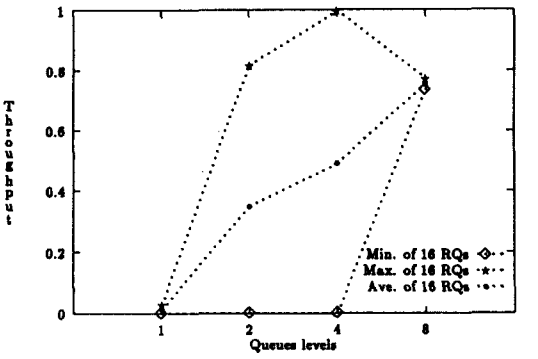

그림 5. 시스템 버스상의 Throughput과 큐와의 관계

Fig. 5. System Bus Throughput Related to Memory Queue.

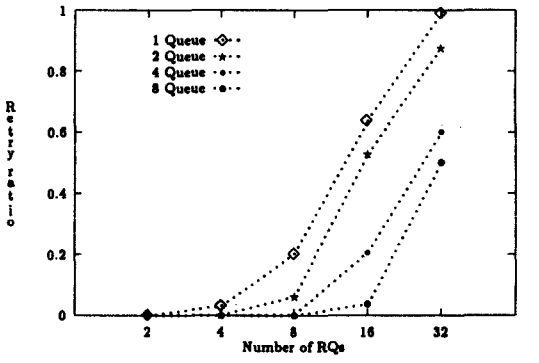

그림 6. 시스템 버스의 재시도와 큐와의 2-RP

Fig. 6. Retry Ratio of System Bus Related to Memory Queue.

그림 6은 시스템 버스상의 재시도에 미치는 영향을

보이고 있다. 입출력 큐들이 재시도를 줄이는 효과를 보이고 있으며 결국 시스템 버스의 유효 대역폭을 증가시킨 것과 같이 더 많은 처리기들을 연결할 수 있도록 해준다.

물론 위의 효과들은 주기억장치의 수를 늘리고 주기억장치간의 인터리빙을 사용함으로서 유사한 결과를 얻을 수도 있다. 그러나 주기억장치는 DRAM의 고집적의 효과로 각 보드당 기억 용량이 매우 크다는 것과[1] 고집적 DRAM의 소자 가격이 높다는 것을 고려할 때 주 기억장치의 수를 늘리는 것은 비용 효과면에서 좋은 방법이 아니다.

Ⅳ. 비동기식 주기억장치의 구현

1. 구현

비동기식 주기억장치는 설계및 구현의 용이성과 개선시의 유연성을 고려하여 크게 세 부분으로 나뉘어 설계된다. 이들은 내부 어레이(Internal Array), 외부 인터페이스(External Interface) 그리고 버퍼들로 구성되는 입출력 큐이다. 내부 어레이는 기억소자 제어기, SEC/DED를 위한 오류 처리기, DRAM 소자 그리고 상태 및 제어 레지스터들로 구성되고 외부 인터페이스는 응답기로서 시스템 버스 인터페이스, 응답 제어기, 타이밍 펄스 발생기와 데이터 전송기등으로 구성된다. 입출력 큐는 여러 종류의 큐들과 그 제어기로 구성되며 내부 어레이와 외부 인터페이스를 분리시킴으로써 주기억장치가 비동기식이 되도록 한다.

그림 4에 나타난 것과 같이 이들 세 부분은 다시 10개의 모듈들로 나뉘며 각 모듈의 기능을 표 2에 나타내었다.

읽기 종류의 요구에 대한 동작은 다음과 같다. 외부 인터페이스 부분, RPAC, 은 주소 정보를 시스템 버스로 부터 받아 분석하여 요구의 종류와 자신에 대한 것인지 판단하여 필요한 정보를 입력 큐에 넣는다. 내부 어레이 부분은 처리 중인 일을 마친 후에 입력 큐가 비어있는가 검사하고 비어있지 않을 경우 그 내용에 따라 적절한 기억 장치에 대한 동작을 제어한다. 즉, DRAM에 대한 읽기 제어를 하고 EDC를 통하여 오류 검사를 진행하도록 한다. 읽기를 마치면 출력 큐를 통하여 읽혀진 데이터를 외부 인터페이스로 보내고 다음 큐를 검사한다. 외부 인터페이스의 또 다른 부분, DREC, 이 큐를 검사하여 시스템 버스의 프로토콜에 따라 데이터를 전송한다. 쓰기 요구가 있는 경우도 읽기의 경우와 유사한 순서로 동작하며 데이터가 DRAM에 쓰여지고 시스템 버스를 통한 전송이 없다는 점이 다르다. 또 특별히 캐시 동작 또는 시스템 버스상의 전송 오류등에 의하여 이미 입력 큐에 입력된 동작을 도중에 취소시킬 수 있도록 별도의 제어 큐를 두고 있다. 이것은 역시 시스템 버스의 파이프라인 방식과 최대한 효과적으로 인터페이스하기 위한 것으로 캐시 응답등을 알 수 있는 파이프라인 단계 이전에 주기억장치의 동작을 시작함으로써 많은 경우의 접근시간을 단축시키기 위한 것이다.

표 2. 비동기식 주기억장치의 구성 모듈
Table. 2. Modules of Asynchronous Memory.

모듈 명	기 능
RPBI	ResPonder Bus Interface 시스템 버스 인터페이스를 위한 신호 송수신기와 래치
TPG	Timing Pulse Generator 정밀제어를 위한 타이밍 펄스 발생기
RPAC	ResPonder Activation Controller 시스템 버스로 부터 요구를 받아 내부로 전달
DREC	Data REturn Controller 내부 어레이로 부터 데이터를 받아 시스템 버스로 전송
IQ/OQ	Input Queue / Output Queue 입출력 큐
MEMC	MEMory Controller DRAM 소자들과 내부 데이터 경로의 제어
REFT	REFresh Timer 리프레시 간격을 결정
EDC	Error Detection & Correction ECC의 생성과 검사
DRAM	DRAM array
CSR	상태 및 제어 레지스터

본 주기억장치의 구현에는 ALTERA의 EPLD가 사용되었으며 이것은 제한된 보드 위에 256MByte의 DRAM을 실장하기 위하여 제어회로의 소자 수를 줄여야 했기 때문이며 동시에 개발 기간도 단축할 수 있었다. 구현에 고려된 소자들은 EPLD외에도 고속 PAL, Crosspoint 소자, Xilinx FPGA, 그리고 게이트 어레이등이 있었으며 소자 선정시 필요한 레지스터의 수, 입출력 신호의 수, 동작 속도, 사용의 용이성, 개발 기간, 그리고 가격등이 주로 고려되었다. 이들 중 고속 PAL은 소자의 수가 많이 필요하고 그에 따라 신호 지연 시간도 더 필요하다는 점에서, 그리고 게이트 어레이는 주기억장치가 타 블록보다 먼저 개발될 필요가 있었으므로 개발 기간상의 문제로 제외되었다. FPGA는 충분한 레지스터와 입출력 신호선을 제공하고 개발 경험도 있다는 장점이 있었으나 고속 동작을 위한 최적의 라우팅을 위해서는 개발 기간을 단축하기 어려웠으며 Crosspoint 소자의 경우는 소자 자체의 속도와 유연성등의 장점에도 불구

1) 고속중형 컴퓨터의 경우 보드당 256 MByte임

하고 재 프로그램이 불가능하여 결국 개발 비용이 많이 필요하여 제외되었다. EPLD는 가격과 개발의 용이성은 우수하나 복잡한 제어 회로의 경우 동작 속도가 떨어진다는 단점이 있으나 표본 회로에 대한 모의 평가를 한 결과 FPGA나 Crosspoint와 비슷한 속도를 얻을 수 있다고 판단되어 선정하였다.

외부 인터페이스는 시스템 버스와 같은 60 ns의 기준 클럭이 사용되었으나 내부 어레이 부분에서는 보다 정밀하고 효율적인 제어를 위하여 30 ns의 클럭과 7.5 ns 간격의 타이밍 펄스들이 사용되었다. 많은 터미네이션 저항들이 신호의 특성 유지를 위하여 사용되었다. 각 모듈들은 Verilog-XL 또는 EPLD 개발 도구를 사용하여 논리적 동작 및 시간적 규격의 만족 여부를 시뮬레이션하였고 최종적으로 전체 주기억장치 블록을 통합하여 Verilog-XL을 이용한 시뮬레이션을 하였다. PCB 작업을 위하여 Allegro를 이용하였으며 약 360개의 소자를 실장하는 10 레이어를 갖는 PCB를 설계, 제작하였다.[11]

2. 시험 및 결과

주기억장치는 다음과 같은 기본 시험 계획에 준하여 시험되었다.

1. 외관 시험 : PCB의 상태 및 부품의 조립 상태와 전원을 시험한다.
2. 기본 시험 : 타이밍 펄스의 특성 및 초기화 시험을 한다.
3. 단독 기능 시험 : 주요 모듈의 기본 동작을 시험하며 이때 주요 신호의 특성을 시험, 조정한다.
4. 블록 기능 시험 : 주기억장치가 수행할 여러 읽기, 쓰기 동작들을 구분하여 또는 조합하여 블록의 기능을 시험한다.
5. 부하 시험 : 장시간 많은 부하를 걸은 상태에서 시험하며 수일간 계속 시험되기도 한다.

위와 같은 시험을 거친 후 본 주기억장치의 동작이 확인되었으며 자체 제작한 시스템 버스 감시기를 통하여 시스템 버스의 사용률을 조사하였다. 펜티엄 프로세서를 실장한 세개의 처리기와 하나의 주기억장치를 시스템 버스에 연결한 상태에서 시스템 버스의 사용률은 평균 7 % 수준이었으며 10 %를 넘는 경우는 관찰되지 않았다. 이때의 workload는 기본 하드웨어 밖에 없는 상태이므로 하드웨어 시험용 프로그램이 사용되었다. 이 시험 프로그램은 다수의 처리기들이 독립적으로 동작하며 시스템 버스상에서 많은 자원 충돌을 일으키도록 구성되어 있다. 중요한 점은 이와 같이 과다한 주기억장치 접근을 요구하는 시험 프로그램을 사용하였을 때도 시스템 버스상에 재 시도가 거의 나타나지 않았다는 점으로 시뮬레이션에서 나타난 바와 같이 비동기식 동작 방식이 시스템 버스의 파이프라인 동작에 매우 효과적이라는 것을 확인 할 수 있었다. 또한 세개의 처리기에 대하여 하나의 주기억장치 즉, 인터리빙이 없는 상태에서도 주기억장치 요구가 효과적으로 처리된다는 것은 비동기식 주기억장치가 매우 가격 대비 효과가 높다는 것이 확인된 것이라고 할 수 있다.

V. 결 론

본 논문에서는 효과적인 다중처리기 시스템을 구성하기 위하여 시스템 버스에서 파이프라인 방식을 지원할 때 시스템 버스의 유효 대역폭을 효율적으로 이용하기 위한 비동기식 주기억장치 구조를 제안하였다. 제안된 구조를 효과를 알아보기 위하여 Verilog 언어를 사용한 시뮬레이션 모델을 작성하여 시뮬레이션하였으며 실제 설계를 위한 구조 분할을 하였다. 제안된 주기억장치의 구현을 위한 설계 내용 및 구현상의 변수들을 평가하여 구현 기술을 선택하였다. 구현된 주기억장치 보드는 사전에 준비된 환경과 시험 계획에 의하여 시험되었고 그 동작이 확인되었다. 또한 부하 시험을 통하여 제안된 구조가 파이프라인 방식의 시스템 버스의 효율성을 높여주는 가격 대비 효과가 높은 구조라는 것을 확인하였다. 즉, 비동기식 주기억장치는 시스템 버스상의 재시도를 대폭 제거하여 시스템 버스의 사용 효율을 높이고 결과적으로 확장성을 증가시킨다. 재시도를 없앤다는 것은 재시도의 경우 필요한 프로토콜상의 오버헤드를 없앤다는 것이므로 평균 접근시간의 개선에도 기여한다고 할 수 있다. 특히 비동기식 구조가 시스템 버스의 블록 전송을 공평하고 효과적으로 지원할 수 있도록 하는 효과를 보이고 있으므로 블록 전송을 필요로 하는 입출력 동작이나 보다 큰 캐시 라인을 지원할 수 있도록 한다.

앞으로의 연구 및 개선 방향으로는 비동기식 구조의 큐에의한 오버헤드를 최소화 할 수 있는 방법을 찾는 것과 내부 어레이의 오류 검사로 인한 오버헤드를 줄이는 것이 있다. 특히 오류 검사를 위한 EDC기능을 제어기능과 통합 구현할 수 있는 ASIC기술이 필요할 것이다. 블록 전송기능과 관련하여서는 캐시 시스템에 대한 모델과 통합 평가하여야 최적의 블록 크기와 그에 맞는 큐의 크기 및 시스템 버스의 기능 보완점등을 추출할 수 있을 것이다. 또한 시스템의 상용화가 진행된 뒤에는 실제 상용의 응용 부하 또는 유사한 벤치마킹 부하를 걸었을 때의 효과를 분석할 수 있을 것이다.

參 考 文 獻

[1] P. Woodbury, A. Wilson, B. Shein and I. Gertner, " Shared Memory Multiprocessors: The Right Approach to Parallel Processing", COMPCON Spring, pp 72-80, 1989.

[2] Shreekant Thakkar, and et. al., " Balance: A Shared Memory Multiprocessor System", Proc. 2nd International conference on super-computing, Vol.1, pp.93-101, 1987.

[3] 윤 용호, "다중처리 시스템: TICOM", IEEE 한국 지부 병렬처리 컴퓨터 워크샵, pp. 25-42, June, 1992

[4] A. J. Smith, "Line Size Selection in CPU Cache Memories", IEEE Tran. on Computers, C-36,9, pp 1064-1075, Sept., 1987.

[5] Susan Eggers, Randy Katz, "The Effects of Sharing on the cache and Bus Performance of Parallel Programs", Proc. ASPLOS III pp. 257-270, Apr., 1989.

[6] J. Archibald, J.L. Baer, "Cache Coherence Protocols: Evaluation Using Multiprocessor Simulation Model", ACM TOCS, pp 273-298, Nov., 1986.

[7] M. C. Chiang, G. S. Sohi, "Evaluating Design Choices for Shared Bus Multiprocessors in a Throughput Oriented Environment", IEEE Trans. on computers, C-41,3, pp 297-317, Mar., 1992.

[8] C.H. Won, J.S. Hahm, K.S. Jeon and W.J. Hahn, "Evaluation of Shared Memory Multiprocessor", JTC-CSCC '90, pp 88-91, Dec., 1990.

[9] A.D. Ki, W.S. Sim and B.K. Park, "Highly Pipelined Bus : HiPi-Bus", JTC-CSCC' 91, pp 528-533, July., 1991.

[10] W.J. Hahn, K. Park, N.J. Park, G.H. Oh, S.W. Kim and S.H. Yoon, "Evaluation of Buffered Memory for Shared Bus with Pipelined Protocol", JTC-CSCC '93, pp 867-871, July., 1993.

[11] 한 우종, 박 경, 박 남진, 오 귀현, 윤 석한, 김 수원, "Design of Buffered Memory for HiPi+Bus" 대한전자공학회 추계종합학술 발표대회 논문지, pp 368-371, Nov., 1993

著 者 紹 介

韓 宇 宗(正會員)

1959年 1月 2日生. 1981年 2月 고려대학교 전자공학과 학사. 1984年 9月 고려대학교 전자공학과 석사. 1990年 3月 ~ 현재 고려대학교 전자공학과 박사과정중. 1985年 1月 ~ 현재 한국전자통신연구소 프로세서 연구실 선임 연구원. 주관심 분야는 마이크로프로세서, 컴퓨터 구조, 병렬처리, 상호 연결망 등임.

金 壽 遠(正會員) 第 27卷 第 4號 參照

현재 고려대학교 전자공학과 부교수